Towards a Scalable Hardware/Software Co-Design Platform for Real-time Pedestrian Tracking Based on a ZYNQ-7000 Device

Zheqi Yu, Shufan Yang, Ian Sillitoe, Kevan Buckley
School of Mathematics and Computer Science University of Wolverhampton, UK
Email: Z.Yu@wlv.ac.uk, S.Yang@wlv.ac.uk, I.Sillitoe@wlv.ac.uk, K.A.Buckley@wlv.ac.uk

***Abstract*—Currently, most designers face a daunting task to research different design flows and learn the intricacies of specific software from various manufacturers in hardware/software co-design. An urgent need of creating a scalable hardware/software co-design platform has become a key strategic element for developing hardware/software integrated systems. In this paper, we propose a new design flow for building a scalable co-design platform on FPGA-based system-on-chip. We employ an integrated approach to implement a histogram oriented gradients (HOG) and a support vector machine (SVM) classification on a programmable device for pedestrian tracking. Not only was hardware resource analysis reported, but the precision and success rates of pedestrian tracking on nine open access image data sets are also analysed. Finally, our proposed design flow can be used for any real-time image processing-related products on programmable ZYNQ-based embedded systems, which benefits from a reduced design time and provide a scalable solution for embedded image processing products.**

Keywords—Object tracking; Open-access; System-on-Chip; Hardware/Software co-design

I. INTRODUCTION

Pedestrian detection and tracking is an important research topic in computer vision with potential applications to autonomous driving, smart CCTV and robotics. However, evaluating the detection and tracking of algorithms on open referenced image data sets is a challenging problem [1]. Due to the difficulty of importing various media formats. Evaluating of new algorithms on an embedded platform is further complicated on resource constrained hardware. In relation to Verilog or VHDL RTL, most programmers have a background knowledge gap relating to the hardware description language. Since there is a long learning curve, meaning low design production, it is tough to justify new algorithms into an unknown system [2].

Current hardware/software co-design usually embrace reuse modular methodologies. The components in hardware/software co-design, called intellectual property (IP) blocks or cores, are typically synthesizable RTL designs (often called soft cores). The concept of soft modular can be carried out at blocks, platforms, or chip levels, and involves making the IP sufficiently general, configurable or programmable for a wide range of applications. Most of the current techniques in hardware/software co-design address only specific elements of the overall problem and there still remains a need to provide a coherent and integrated method, especially a software developer-friendly environment to address the issues of integration, synchronisation, communication and scheduling.

Xilinx Inc., provides an HLS software tool for their FPGA products [3]. Current high-level synthesis tools enable an automatic synthesis from untimed specifications using a high-level language, such as C/C++ programming language [3] to a low-level cycle-accurate register transfer level (RTL) specification for FPGA developments. According to the Xilinx reference design in the latest Xilinx software development kit for Vivado, an automated HLS flow allows designers to specify design functionality in high-level programming languages for customised hardware logic on their ZYNQ SoC FPGA devices [4]. Vivado also provides effective methods for controlling complex IPs to improve design productivities. However, adopting HLS design flow for a common FPGA design is still a challenging task in terms of configuring a robust compilation process and platform-based model, and providing the availability of application-specific design templates.

In this work, we describe a scalable hardware and software co-design platform that is easy to use and allows the ready investigation and comparison of various computer vision algorithms. As a case study, we demonstrate an approach to developing a histogram of oriented gradients (HOG) algorithm based pedestrian tracking system using a Xilinx ZYNQ-7000 chip. We provide details of the architectures and all the most important steps to give examples of integration and evaluation for pedestrian tracking implementation. The proposed design flow greatly reduces the hardware/software development time, and simplifies the design cycles.

II. RELATED WORK

FPGAs are commonly used in machine vision applications. It is used to ensure vision systems be able to run at high speed and at real-time [2]. Many machine vision applications have been reported with carefully considered hardware complexities. For instance, a visual tracking system is used an optimised optical flow algorithm for VLIW DSP architectures, whilst a particle filter based approach for multiple object tracking implemented on an FPGA [5]. However, the power consumption and cost of the hardware platforms are too expensive for embedded systems to allow those systems are developing into practical products.

Prisacariu and Reid [6] use four GeForce GTX 295

graphics cards in SLI (Scalable Link Interface) configuration to run HOG algorithm, and achieve a frame rate of 10 frames per second for 1920 x 1080 images. Although down-scaling the input image resolution can increase the processing speed, the power consumption of this new platform is difficult to meet the requirement on some portable devices [7].

Few attempts have been made for embedded image processing with the less computationally expensive algorithm. Kadota et al [8] implemented a partial pedestrian tracking solution based on the HOG algorithm, which achieves 30 FPS in 640 x 480 resolution video at 127.49 MHz. However, their approach was a highly optimised hardware implementation with less flexibility for future extension. Hsiao et al [9] proposed an ARM and FPGA co-design to implement the entire HOG algorithm. Their design supported multiple classifiers (i.e, SVM and AdaBoost) and was able to process 15 FPS whilst detecting 1075 objects per second. In their design, they chose the tegra2 1GHz CPU (ARM) and Spartan-6 XC6SLX150T 192MHz (FPGA) to self-design PCB, which communicated through an ARM generic memory interface (GMI) data bus link to the FPGAs. However, due to the limitation of customised hardware platform, their system was lacking the generality.

III. System Overview And Design Flow

We develop a scalable hardware/software co-design platform, aiming to encourage software engineers to treat FPGAs as programmable computing resources while achieving phenomenal performance without directly developing hardware high description language. We demonstrate how to use C or C++ programming language to support efficient hardware/software co-design and improve prototyping process by implementing software proofed algorithms directly into hardware/software co-design platforms with highly parallel FPGA platforms.

A. Design flow

Comparing to traditional hardware design flow, our developing flow includes kernel configuration and cross-platform compilation, customised bootloader, and finally, IP core integration processes that generate the hardware description file using an HLS (High Level Synthesis) step. In addition, we design a new driver based on the V4L2 (video for linux 2) framework and it can trigger interrupt handling that allows the VDMA (video direct memory access) commence transferring data to PL (Programmable Logic). The VDMA is a special direct memory access method with the configuration of dual-buffering and multi-buffering mechanisms. So it can be efficiently communicated with DDR3 while converting AXI Stream data streams into memory map format or converting memory map data back to AXI Stream data.

Fig. 1 shows a co-design operating system design flow. An interrupt (in the VDMA) handling method is used to process request from the VDMA IP memory map (MMAP) and enables a user space memory address to operate associated devices. The interrupt allows the application program to read or write using a direct memory access method. When a program uses the frame buffer, it needs to map the frame buffer to the on-chip processor's address space. Take

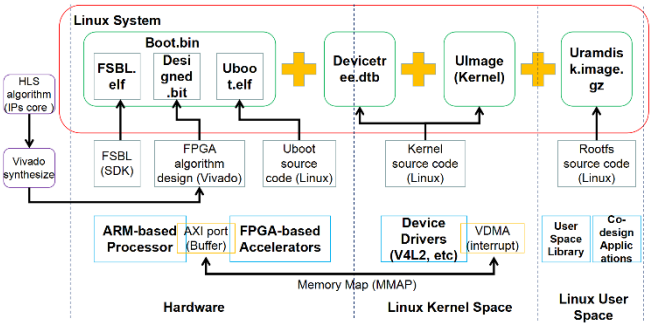

Fig. 1. The design flow of the ZYNQ Co-design Linux System.

advantage of the flexible buffering access; any video stream can be imported into our system. An MIPI CSI-2 interface is used to process single and dual stream input data, then display the RAW output frame buffer in OV490 camera driver. Furthermore, a particular memory mapping scheme is developed to allow hardware IPs integrated with the embedded Linux system. As it shown in Fig. 1, the BOOT.bin file includes First Stage Boot Loader FSBL.elf, design wrapper.bit (from the Vivdao project) and U-boot.elf (from the U-BOOT source code). Using our ported embedded Linux system, many basic systems functions and application libraries, such as QT and OpenCV libraries can be cross-compiled and executed on the targeted board. This embedded Linux system framework based on open sourced Linux project, its technical support by the Xilinx Inc, and also can be upgraded Rootfs to gets Ubuntu and Debian system. Thus, the project developing time is saved by only modifying configuration file BOOT.bin with customised IPs without redesign the whole system.

B. System-on-Chip Design Hierarchy

The project presents the standard flow for the algorithm development, validation and optimisation using the Xilinx Synthesis Tool, which enables an HW/SW implementation. It uses some primary IP cores are a processing system, an HDMI module, two axi protocol converter modules, an axi vdma module, an xlconcat module, an axi interconnect module, a proc sys reset a module and customised image filter IP core that completes colour-to-grey convert function. Since the axi interconnect IP cannot be directly connected to the axi stream interface, a memory map is built for the axi streams, which enables image data to transfer through the direct memory access (DMA) of the memory map (MMAP) to the axi stream. We use a VDMA IP core to provide a stream for video data. The modules axi interconnect and axi protocol converter are imported from Xilinx IP core library.

Vivado HLS software generates the image filter processing module. Although this is a relatively less computationally expensive function, our purpose is to demonstrate the facility to explore design space for any given image processing applications and to increase design module reusability.

C. Adapted HOG Algorithm in the Software Design Flow

In this work, we demonstrate how to use a C-language enabled design flow to support efficient hardware/software codesign, and reduce prototyping design time by implementing software applications directly onto hardware/software co-design. This design methodology

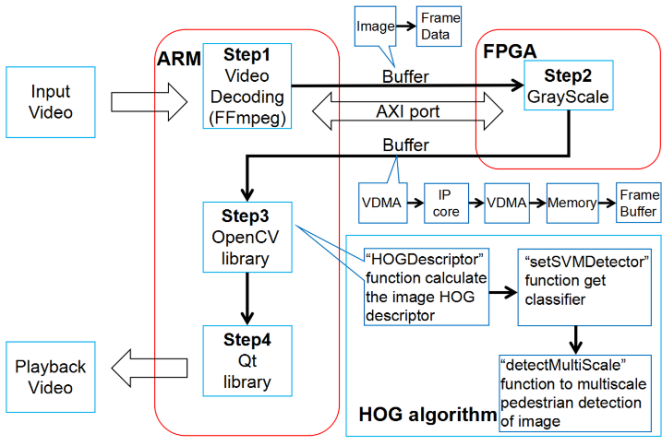

Fig. 2. HOG algorithm's working flow in co-design.

Fig. 3. Prototype platform using Xilinx ZYNQ-ZC702 board.

enables software programmers to execute and test the same C algorithms in desktop PC environments (port C algorithms by the HLS) and implement systems using embedded processors with highly parallel FPGA platforms.

The histogram of oriented gradients (HOG) classifier is a classical algorithm application on pedestrian detection used by the method of feature extraction and machine learning [10]. The HOG feature descriptor was utilised for the distribution of intensity gradients to describe local object appearance and shape within an image [10]. Four steps are included in HOG algorithm. First the local histograms are used to calculate a measure of the intensity of a larger region of an image, called a block, and then the value is used to normalise all cells within the block. During the HOG algorithm processing for detecting pedestrians, the following features are included: coarse spatial sampling, fine orientation sampling and strong local photometric normalisation. Those steps permit the individual body movement of pedestrians to be ignored as long as they maintain a roughly upright position.

Fig. 2 shows an integrated HOG co-design flow. In Step 1, the video source is decoded by the FFmpeg function, and then raw data is transmitted to FPGA. Step 2 shows the FPGA HLS workflow. Through the VDMA, the frame data are sent to HLS IP for processing, and the GrayScale IP returns the processed result by the VDMA. The memory map function is used to allow data to store in the memory and finally transfer it to the frame buffer. This step is important for image processing flow in the PL. The FPGA only allows the image transmission by the stream format that was converted by the VDMA, so the VDMA needs to generate requests to use the real physical address and leave reserve space [4]. If the system is without the driver control for the VDMA, it needs to manually set the hardware control that sets the hardware address for the memory map and locate the VDMA control logic registers. The memory mapping scheme allows developers to use buffers for data interaction without setting any address. In this project demo, the colour-to-grey conversion function uses the HLS design to get GrayScale IP core.

In step 3 is the details of the HOG algorithm workflow. We use ARM to directly call the HOGDescriptor algorithm function in the OpenCV library and cooperate with the FPGA colour-to-grey conversation function. This step is followed by Dalal and Triggss work [10]. After the current frame is divided into the cell units, the gradients or edge of the orientation histogram of each cells unit is calculated. Then, the orientation histograms of each cell are combined to construct a feature descriptor. The setSVMDetector function is used to achieve coefficient assignment to classify different characteristics of the HOG SVM model. After HOG descriptors are calculated, the final step in pedestrian tracking is to feed the descriptors into a recognition system based on supervised learning. In this project, the HOG classifier based on SVM is trained with the INRIA training set. The amount and type of resources required by the Xilinx ZYNQ-ZC702 board for HOG and SVM-based pedestrian tracking are 29% of LUT and 14% of BRAM for the GrayScale IP. With this design, the estimated clock cycle is 11.12ns (worst case) and 10ns (target case) in the Vivado HLS design. It should be noted that the hardware utilisation forecast from Vivado HLS changes after RTL synthesis (generated hardware description language). It is necessary to maintain a margin for hardware device choice in order to save enough design margin.

IV. Implementation And Results

In this work, we propose to use a general solution and prototyping design, which allows any software developers to quickly validate and prototype image applications in an environment very close to the final implementation with a real-time execution on an embedded hardware architecture. For other boards, the development will be the same concept for changing synthesis steps.

Fig. 3 shows the prototyping environment that we use the Xilinx ZYNQ-ZC702 board to demonstrate project. The pedestrian tracking algorithms are implemented in an FPGA-based SoC platform ZYNQ-7000 series development kit produced by the Xilinx [13]. All programmable system-on-chip (SoC) Xilinxs ZYNQ-7000 series is the first-generation production of an extensible processing platform (EPP) which is supported by an HLS design flow. It has various features such as software/hardware co-simulation and I/O programmability [14]. The ZYNQ platform has a robust built-in ARM processor-based processing system (PS) and 28-nanometer Xilinx programmable logic (PL). It is a great

innovation compared to the previous FPGA products usually designed by PLs with some optional MCU extensions (PLCenter Architecture).

D. Hardware Platform

The hardware platform uses Xilinx's ZYNQ-7000 series SoC. The advantage of this platform is the chip integration of the ARM (processing system, PS) and FPGA (Xilinx programmable logic, PL), which provides a unique hardware/software co-design platform [13]. The ARM processing system controls interfaces of peripherals (HDMI and USBs) and provides communication between programming logics and software applications using a processor configuration access port [15]. In ZYNQ-7000 chips, the bus interfaces between the PS and the PL to exchange data use AXI 3.0 bus protocol, including two AXI GP interfaces, four AXI HP interfaces and one AXI ACP master port interface [13]. As is shown in Fig. 3, the ZYNQ-ZC702 board is connected to an LCD monitor, and a laptop is used for monitor Linux system running on ARM processors on the board. The image datasets are preloaded into the SD card for processing.

The Visual Benchmark data set is adopted to evaluate our hardware/software co-design for the pedestrian tracking [16]. We adopt nine sequences (Human2 (480 X 640), Human4 (640 X 480), Human7 (320 X 240), Human8 (320 X 240), Human9 (320 X 240), Singer1 (624 X 352), Skating1 (640 X 360), Trans (640 X 332) and Woman (352 X 288)) to evaluate the processing speed and pedestrian detection precision.

E. Experiment Setup

The data set has 100 videos that capture different objects. In this project, we focus on the Human object. Nine video sequences were chosen to test real-life scenarios according to illumination variation (IV), scale variation (SV), occlusion (OCC), deformation (DEF), motion blur (MB), fast motion (FM), in-plane rotation (IPR), out-of-plane rotation (OPR), background clutter (BC) with different resolutions from QVGA (320 x 240) to VGA (640 x 480). At the same time, each video has multiple features that including different scenes. The most interesting video is Trans whose object is Transformers, but the dataset authors still classify it as human. The test uses two different SVM rates (1.3 and 1.9) controlled by the "detectMultiScale" function [11]. In this paper, a value of 1.3 and 1.9 have used after extensive trials that it the regularisation coefficient [12] is important since this parameter controls the degree of over learning. Thus, a small coefficient value allows a large separation margin between classes, which reduces over learning and improves generalisation.

F. Hardware Utilization

The following chart (Table 1) lists the amount and type of resources required by the Xilinx ZYNQ-ZC702 board for HOG- and SVM-based pedestrian tracking. It illustrates hardware utilisation without hardware optimisation (only using the Vivado HLS software built-in optimisation function). Due to the simplicity of the colour-to-grey conversion function, this design only used 29% of LUT and 14% of BRAM. Although the actual range of the clock cycle is a targeted clock cycle plus or minus 12.5%, the current setup can prevent unknown network latency due to the signal was shown. With this design, the estimated clock cycle is 11.12ns (worst case) and 10ns (target case) in the Vivado HLS design. It should be noted that the hardware utilisation forecast from Vivado HLS changes after RTL synthesis (generated hardware description language). It mainly uses LUT. Hence, it is necessary to maintain a margin for hardware device choice in order to save enough design margins.

TABLE I. REPORT OF UTILIZATION

Resource	*Utilization*	*Available*	*Utilization %*
LUT	15167	53200	28.51
LUTRAM	2128	17400	12.23
FF	19948	106400	18.75
BRAM	19.50	140	13.93
DSP	19	220	8.64
IO	20	200	10.00
BUFG	5	32	15.62

G. Pedestrian Detection Performance

Wu et al. [16] proposed benchmarks for testing visual pedestrian tracking: precision rates, success rates and robustness. In this work, we focus on testing the real-time image processing performance instead of the tracker performance. Therefore, we only analyses precision rates and success rates. Precision rates indicate the ratio of frames with average centre location error below a threshold to the ground truth from the visual tracker benchmark [16]. In the HOG algorithm results, the project got the object bounding box location for X and Y position both at the left bottom and top right corner. Also, according to the calculation, there is bounding box of four numerical to left bottom X, Y position, and the box of width and height. Finally, the bounding box of four numerical as the matrix table are placed into the visual tracker benchmark, so that the figures will show the algorithm precision rates and success rates.

In the VGA resolution videos, the best result is from the Human2 sequence (Fig. 4A and Fig. 5A). At the same time, in the QVGA resolution videos, the best result is from the Human7 sequence (Fig. 4B and Fig. 5B). Since the size of pedestrian objects in the video sequences Singer1, Human8 and Human4 are 10% smaller than the size of detection windows (64 pixels by 128 pixels), those three video sequences are not shown in Fig. 4 and 5. Moreover, the results of video sequences Human2 and Human7 have the same increase rates for all of the four platforms. In the QVGA resolution videos, the co-design is normally better than the single ARM software design due to FPGA hardware acceleration advantage. Besides, in the VGA resolution videos, those advantages have been overtaken by the decoding overhead during the processors processing of video stream decoding. We also analyses the success rates, which indicate the ratios of successful frames at the thresholds varied from 0 to 1, comparing them to ground truth. Resultant success plots are shown in Fig. 5, which shows that our implementation is superior in both scale and aspect ratio adaptability during various video sequences. In the results, ARM13 indicates the use of only the ARM version running for SVM rate 1.3, Co--

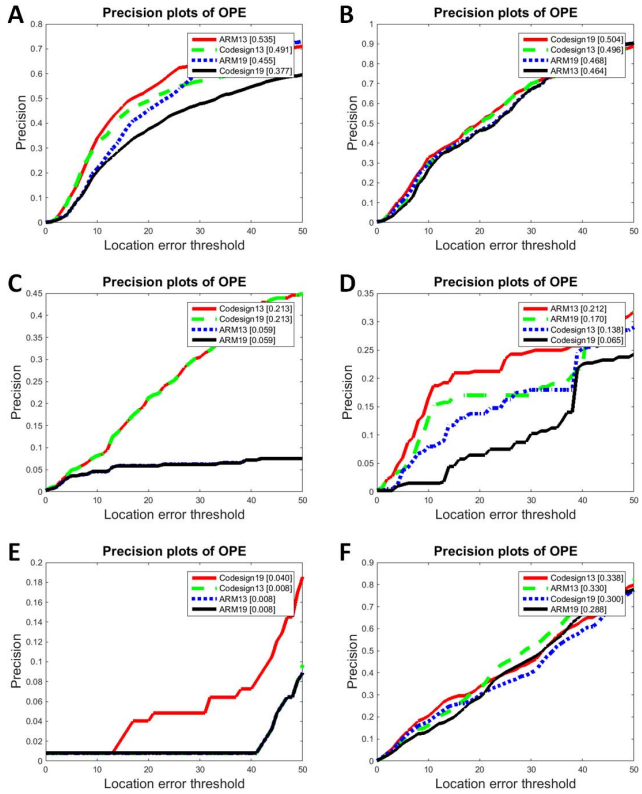

Fig. 4. Precision rates of A. Human2, B. Human7, C. Human9, D. Skating1, E. Trans, and F. Woman.

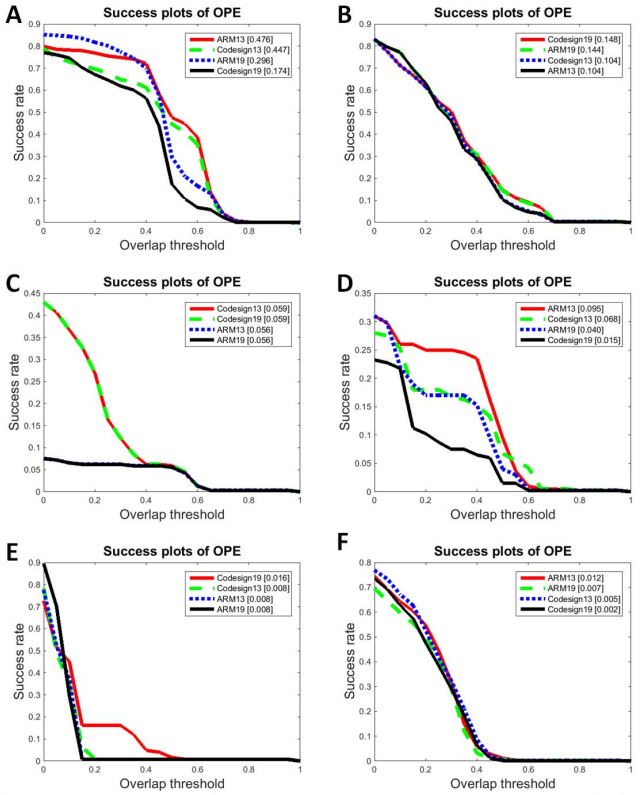

Fig. 5. Success rates of A. Human2, B. Human7, C. Human9, D. Skating1, E. Trans, and F. Woman.

design13 indicates the use of the ARM and FPGA co-design version running for the SVM rate 1.3, ARM19 indicates the use of only the ARM version running for the SVM rate 1.9, and Co-design19 indicates the use of the ARM and FPGA co-design version running for the SVM rate 1.9. The precision and success rates are three decimal places following the version name (1.000 equals 100).

H. Real-time tracking performance

Fig. 6 shows the correlation of image resolution with computing speeds in QVGA and VGA formats, each of which has two SVM rates and two implementations: ARM core software-only and hardware/software co-design. As Fig. 6 illustrates, the single approach of the ARM achieves faster frame rates than the FPGA co-design, although FPGA has a better acceleration ability. This is because, in the test, we only implement the colour-to-grey function as less than 10% of the computation cost used in the HOG-based pedestrian tracking approach [6]. The acceleration speed is hardly able to overcome the communication overhead between hardware (FPGA) and software (ARM core). However, in the test, the slowest frame rate of video sequence Human2 is just 0.493462 frame per second (FPS) using the co-design approach with SVM rate 1.3. The best FPS result is the Human9 video sequence, which is 4.46808 FPS using the ARM-core software-only approach with SVM rate 1.9. Those results are much better than other visual tracking methods (compared with included data sets algorithm results) using the software only method [16]. Also, the processing speed is a negative correlation for the video resolutions because the VGA resolution is four times larger than the size of the QVGA, and

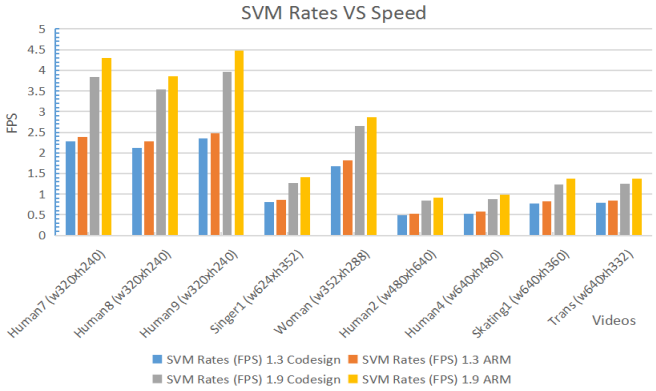

Fig. 6. Relationship of resolutions and speed.

the processing speed for the VGA resolution video is about a quarter of the QVGA speed. The real-time processing speed relies on the video format due to the software only video decoding method. In the future, video decoding processes can be implemented using FPGA, and the pedestrian tracking speed can be improved.

In this work, we propose to use a hardware/software co-design, which allows any software developers to quickly validate and prototype image applications in an environment very close to the final implementation with a real-time execution on embedded hardware. In general, libraries in hardware/software co-design system need to be recompiled. For instance, in tradition High-level synthesis (HLS), it includes three steps that allocation, scheduling and binding [17], and then Xilinx SDK develops a method which involves

a lot of recompilation and redevelopment in FPGA interfaces to on-chip processors. Hence a lot of repeated developing task often happens. However, using our co-design platform, hardware IPs can be used as a device through a DDR shared memory scheme [18].

TABLE II. DEVELOPMENT TIME COMPARISON

Functions	**Design Methods**	
	Xilinx SDK (minutes)	***Our Linux-based H/S co-design (minutes)***
QT library	Not include	~10
OpenCV library	~40 (limited version)	~10
Video CodeC	Only Camera and Photo	~10
Hardware control	~60	~10

In Table 2, we estimate the developing time using Xilinx SDK according to Xilinx White papers [4] and our linux-based co-design platform. Using the Xilinx SDK developing method, all libraries need to be recompiled and separated buffers are required to be re-designed for each DMAs. According to the complexity of libraries, an approach that using Xilinx SDK takes around 60 mins for a rebuild since project hardware control depends on the address setting, and the SDK project should build it every time. While using our linux-based hardware/software (H/S) co-design, no recompilation is needed and 10 mins will be used to copy cross executable file only.

V. CONCLUSION

Traditional hardware/software co-design flow becomes a barrier for a rapid product design. We present a scalable HLS design flow-enabled software and hardware co-design platform with a detailed demonstration of the capability of evaluating real-time image processing algorithms. Followed up our memory map design method, our FPGA-based acceleration modules can work with the operating system in embedded processors. This work brings up the opportunities for any other acceleration modules to be integrated into a standalone computer image processing product. This strategy also creates a scalability feature that any accelerated modules in FPGA to be extended to handle an increased computing load in an any large system. Meanwhile, this project is user-friendly and is easily modified for other form of hardware/software co-design. In future, our scalable-codesign framework can be improved using the play-and-plug GUI interface to enable seamless integration, automatically compile and implementation in any programmable system-on-chip platforms.

REFERENCES

[1] A. W. Smeulders et al, "Visual tracking: An experimental survey," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 36, (7), pp. 1442-1468, 2014.

[2] B. Senouci, I. Charfi, B. Heyrman, J. Dubois, and J. Miteran, “Fast prototyping of a soc-based smart-camera: a real-time fall detection case study,” Journal of Real-Time Image Processing, pp. 1–14, 2014.

[3] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, “High-level synthesis for fpgas: From prototyping to deployment,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 4, pp. 473–491, 2011.

[4] Xilinx, “Zynq-7000 ap soc technical reference manual,” UG585, Tech. Rep. v1.10, 2015.

[5] J. Jin, S. Lee, B. Jeon, T. T. Nguyen, and J. W. Jeon, “Real-time multiple object centroid tracking for gesture recognition based on fpga,” in Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication. ACM, 2013, p. 80.

[6] V. Prisacariu and I. Reid, "fastHOG-a real-time GPU implementation of HOG," Department of Engineering Science, vol. 2310, (9), 2009.

[7] D. B. Thomas, L. Howes and W. Luk, "A comparison of CPUs, GPUs, FPGAs, and massively parallel processor arrays for random number generation," in Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 2009, .

[8] R. Kadota, H. Sugano, M. Hiromoto, H. Ochi, R. Miyamoto, and Y. Nakamura, “Hardware architecture for hog feature extraction,” in Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP’09. Fifth International Conference on. IEEE, 2009, pp. 1330–1333.

[9] P.-Y. Hsiao, S.-Y. Lin, and S.-S. Huang, “An fpga based human detection system with embedded platform,” Microelectronic Engineering, vol. 138, pp. 42–46, 2015.

[10] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), vol. 1. IEEE, 2005, pp. 886–893.

[11] T. Dai, Y. Dou, H. Tian, and Z. Huang, “The study of classifier detection time based on opencv,” in Computational Intelligence and Design (ISCID), 2012 Fifth International Symposium on, vol. 2. IEEE, 2012, pp. 466–469.

[12] C. J. Burges, “A tutorial on support vector machines for pattern recognition,” Data mining and knowledge discovery, vol. 2, no. 2, pp. 121–167, 1998.

[13] Xilinx, “Zynq-7000 all programmable soc zc702 base targeted reference design,” UG925, Tech. Rep. v9.0, 2015.

[14] M. Ruan, H. Guo, and L. Qin, “Wireless base station zuc block cipher implementation on zynq-7000 ap soc,” 2013.

[15] R. Dobai and L. Sekanina, “Image filter evolution on the xilinx zynq platform,” in Adaptive Hardware and Systems (AHS), 2013 NASA/ESA Conference on. IEEE, 2013, pp. 164–171.

[16] Y. Wu, J. Lim, and M.-H. Yang, “Online object tracking: A benchmark,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2013, pp. 2411–2418.

[17] A. Canis et al, "LegUp: High-level synthesis for FPGA-based processor/accelerator systems," in Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 2011.

[18] R. Nane, V. M. Sima, C. P. Quoc, F. Goncalves, and K. Bertels, “Highlevel synthesis in the delft workbench hardware/software co-design toolchain,” in Embedded and Ubiquitous Computing (EUC), 2014 12th IEEE International Conference on. IEEE, 2014, pp. 138–145.